Attorney's Docket No.: 13913-111001 / 2003P00324 US

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: DEVELOPING USER INTERFACES WITH VIEWS

APPLICANT: BJOERN GOERKE, JENS BAUMGART, JENS ITTEL, MARKUS CHERDRON AND STEFAN BECK

MAILING BY EXPRESS MAIL

Express Mail Label No. EV 321 387 331 US

September 30, 2003

Date of Deposit

DEVELOPING USER INTERFACES WITH VIEWS

BACKGROUND

The present invention relates to electronic data processing in general, and particularly to application programming.

Applications can be developed using various architectures, including, for example, a model-view-controller (MVC) architecture. The MVC architecture breaks an application into three separate parts –models, views, and controllers. Each model can have multiple views, where each view displays information about the model to a user. A controller of the model receives events, for example, raised by a user interacting with a view to manipulate the model. Each model can have multiple controllers, and a controller can relate to multiple views. The models and the controllers typically include application code. When changes occur in a model, the model updates its views. Data binding is used for data transport between the view and its model or controller. For example, a table view can be defined to display data of a corresponding table that is stored in the model or controller. The table is used as the data source for the table view (data binding). For example, the table view can be replaced by a further view, such as a graph view, that binds against the same table. In this case, the further view displays the table data without changing anything in the controller or the model.

Application development is often divided into two general stages: design time and runtime. Design time can include designing the views of an application (including the layout of the user interface (UI) elements in each view), modeling of the application flow (including the selection of the views to displayed), designing one or more models, and creating and editing other application elements, such as controllers and contexts. Design time can also include the binding of UI elements within the views to data sources that are defined in a data type repository.

Information created during the design time can include application metadata. Application metadata can be stored in a metadata repository, and used as input to the runtime process. During the runtime process, the application metadata can be used to generate the actual runtime code of an application. In some implementations, the application metadata is

platform independent, and the generated runtime code is platform specific. The runtime code can be executed in a runtime environment that provides a general framework for running applications. For example, a runtime environment can provide services for deploying and maintaining applications, as well as features such as a caching mechanism that can be used to improve performance, and automatic input assistance and default error handling that is based on the declared application metadata.

Regardless of which architecture is used, it is often desirable to structure an application (including, for example, the models, views, and controllers that make up an MVC application) into reusable entities or components. The reusable components can be embedded by the application, or by another reusable component.

SUMMARY OF THE INVENTION

In general, in one aspect, the invention provides methods and apparatus, including computer program products, for developing application user interface through the use of views. The techniques include receiving user input specifying a view composition of two or more views and one or more navigation links, and storing the view composition in a repository. Each view includes a layout of at least one user interface element. Each navigation link specifies a transition from a first view having an outbound plug to one or more second views each having an inbound plug, where each navigation link connects the outbound plug to each of the inbound plugs. The view composition specifies a layout of the views.

Advantageous implementations of the invention include one or more of the following features. The user interface elements can include one or more of input user interface elements, view user interface elements, and container user interface elements. The layout can include a pre-configured pattern of user interface elements. Receiving user input specifying the view composition can include receiving user input specifying a layout of the views using view containers or specifying a layout of the views using a predefined number of view containers in a predefined layout. Receiving user input specifying the view composition can include receiving user input modifying the properties of the user interface elements included in the views. The views can be associated with application content. User input specifying

the view composition can be generated using interface controls provided in at least one graphical user interface.

In another aspect, the invention provides methods and apparatus implementing techniques for developing application user interface through the use of views, where a view composition is received at runtime, the view composition having two or more views and one or more navigation links. Each view includes a layout of at least one user interface element. Each navigation link specifies a transition from a first view having an outbound plug to one or more second views each having an inbound plug, where each navigation link connects the outbound plug to each of the inbound plugs. Runtime code is generated to process the navigation links based on the received view composition, such that a call to an outbound plug at runtime results in those views being displayed that have inbound plugs connected by navigation links to the called outbound plug.

Advantageous implementations of the invention also include creating a new view composition at runtime.

In another aspect, the invention provides a system for developing application user interface through the use of views, including means for receiving user input specifying a view composition of two or more views and one or more navigation links, and means for storing the view composition in a repository. Each view includes a layout of at least one user interface element. Each navigation link specifies a transition from a first view having an outbound plug to one or more second views each having an inbound plug, where each navigation link connects the outbound plug to each of the inbound plugs.

Advantageous implementations of the invention also include a view composition that includes nested views.

In another aspect, the invention provides a data structure for representing a visual interface for a computer program, including a view composition having multiple views and one or more navigation links. Each view includes a layout of at least one user interface element. Each navigation link specifies a transition from a first view having an outbound plug to one or more second views each having an inbound plug, where each navigation link connects the outbound plug to each of the inbound plugs. One or more of the views can be nested.

Advantageous implementations of the invention include one or more of the following features. The views can be nested using view containers or a predefined number of view containers in a predefined layout. The view composition can be represented in XML.

The invention can be implemented to realize one or more of the following advantages. The visual interface for an application is designed using visual interfaces defined by components embedded by the application. Components embed other components and visual interfaces defined by the embedded component are used by the embedding component. A visual interface embeds one or more views, and navigation links are used to define transitions between the views. Because components can be reused, the views associated with the components can also be reused. View compositions can be defined at design time and used to generate view assemblies at runtime. New view assemblies can be created dynamically at runtime through the runtime modification of view compositions. One implementation of the invention provides all of the above advantages.

The details of one or more implementations of the invention are set forth in the accompanying drawings and the description below. Further features, aspects, and advantages of the invention will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a system to allow a user to develop a visual interface of a software application, and present the visual interface at runtime.

FIG. 2 is a block diagram of a view.

FIG. 3 illustrates a visual interface with multiple views that are linked together using navigation links.

FIG. 4 is a flow diagram illustrating a method for designing a visual interface for an application using views.

FIG. 5 is a flow diagram illustrating a method for designing a visual interface for embedded components.

FIG. 6A illustrates a pre-defined layout tool for T-Layout.

FIG. 6B illustrates a pre-defined layout tool for T-Layout 90°.

FIG. 6C illustrates a pre-defined layout tool for T-Layout 180°.

FIG. 6D illustrates a pre-defined layout tool for T-Layout 270°.

FIG. 6E illustrates a pre-defined layout tool for Grid-Layout.

FIG. 7 illustrates a visual interface of a sample shopping application.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

FIG. 1 illustrates an example environment for developing a visual interface of a software application, and presenting the visual interface at runtime. The system includes a development framework 105 and a runtime framework 120. An application developer 100 uses the development framework 105 at design time to specify the visual interface, and stores the visual interface in a repository 110. At runtime, the runtime framework 120 retrieves the specified visual interface from the repository 110 and presents it to an application user 115.

A visual interface of a software application is made up of one or more views arranged in a specific layout. FIG. 2 is a block diagram of a view. A view 200 specifies a layout of at least one user interface element (UI) element 205, and a view area. UI elements in a view can include buttons, labels, and menus. The view area defines the area to be occupied by the view 200 in a visual interface embedding the view 200. The UI elements included in the view 200 can include Input UI elements, View UI elements, and Container UI elements. An Input UI element is used to receive input from the user, e.g., a drop down menu, an input field, or a table UI element. A View UI element is used to display application data, e.g., an image view, a text view, or a caption. A Container UI element, described below, is used to include other views and UI elements, e.g., a scroll container UI element having a scroll bar, or a container UI element specifying a layout for included views.

The visual interface can have more than one view; generally, only some of the views are visible at any time. Which views are visible in the visual interface can change, e.g., the views that are visible can change in response to input from the user. Navigation links are design time constructs made available by the development environment for use by an application developer to specify transitions between the views. Each view has an inbound plug 220 and an outbound plug 225. At design time, each navigation link between an inbound plug 220 and an outbound plug 225 establishes a potential transition from the view with the outbound plug 225 to the view with the inbound plug 220. A developer specifies a

transition from a first view to a second view by connecting the outbound plug 225 of the first view to the inbound plug 220 of the second view. The navigation links are processed at runtime to cause the view transitions specified at design time. At runtime, the application calls the outbound plug of the first view to cause a transition from the first view to the second view.

Each inbound plug 220 includes an application specific event handler, and calling the inbound plug results in running the event handler for the inbound plug 220 before displaying the view 200 corresponding to the inbound plug 220. Navigation links are typically processed in a runtime framework by calling all the inbound plugs 220 connected to an outbound plug 225, by navigation links, when the outbound plug 225 is called. The event handler for an inbound plug 220 can call the outbound plug 225 for the view corresponding to the inbound plug to cause other views connected to the outbound plug 225 to be displayed. The application can use the event handler for the inbound plug 220 to initialize the corresponding view, e.g., the corresponding view can be initialized based on why the view is being displayed.

The view 200 can have an associated view controller that includes the event handlers associated with the inbound plug. The view controller also contains event handlers for the UI elements in the view as well as the presentation logic for the view.

The application or a reusable component can specify any number of views at design time, any of which can be displayed at runtime. The set of views that can be displayed, for the application or the component, is referred to as the view composition. A view assembly is the set of views that are actually displayed at runtime. The view assembly, for an application or a component, consists of views in the view composition that selected for display at a certain point in time. When a navigation link is processed at runtime, a view in a current view assembly may be replaced by one or more destination views from the view composition.

FIG. 3 illustrates a visual interface with multiple views that are linked together using navigation links. Each navigation link connects an inbound plug to an outbound plug. The view area 300 includes three views 305, 310, and 315, of which view 305 is currently displayed in the view area 300. View 305 has inbound plug 315 and outbound plug 320. View 310 has inbound plug 325 and outbound plug 330. View 315 has inbound plug 335

and outbound plug 340. Outbound plug 320 is connected to inbound plug 325 by a navigation link 345, and outbound plug 320 is connected to inbound plug 335 by a navigation link 350. At runtime, calling the outbound plug 320 when view 305 is displayed in the view area 300 causes views 310 and 315 to be displayed in the view area 300 instead of the view 305. In addition, if any application specific event handlers associated with the inbound plugs 325 and 335 are also called.

Applications can make use of components that contain view compositions. Components can embed other components, such that a first component can interact and make use of a second, embedded, component. The view composition of the first component can include views of the second component. Similarly the view composition of the application can include views of the components used by the application. In addition, an application developer can design application specific views that are part of the application's view composition.

The design time environment and the runtime environment provide a special view called the empty view. The empty view contains no UI elements, has no associated view controller, and has only one inbound plug. Application-specified views and views specified by embedded components contain a layout of at least one UI element. At design time the empty view is used to fill portions of the view area where no view is to be displayed. At runtime, if a component view is made visible (e.g., by the navigation links connected to its inbound plug), and the component has not been instantiated yet, the empty view is displayed instead of the component view. In one implementation, the empty view is substituted for the component view by the runtime framework. For example, in an application for taking sales orders, a view area may be set aside for showing a view associated with a shopping cart component (a reusable component that maintains a list of items to purchased by a customer). If the shopping cart component has not been initiated yet, e.g., because the customer has not selected any items, the portion of the view area associated with the shopping cart is filled with empty space by displaying an empty view.

FIG. 4 is a flow diagram illustrating a method for designing a visual interface for an application using views. An application developer specifies the view composition for the application (step 400). Specifying the view composition includes specifying the views

included in the view composition, and specifying the navigation links between the views. Specifying the view composition also includes specifying a layout of the views in a visual interface. Specifying a layout of the views includes specifying groups of views occupying a view area, where only one of the views in a group is visible at any time. The views, navigation links, and the specified layout define the view composition. The view composition is stored in a repository (step 405), in an XML representation, for example.

FIG. 5 is a flow diagram illustrating a method for designing a visual interface for embedded components. Views specified by an embedded component can be embedded by an application or by another component. One or more component windows are defined for the embedded component (step 500), and component views to be included in each component window are specified (step 505). The component views specified in step 505 can be component views defined by the embedded component, or views defined by other components. A layout of the component views for each component window is specified (step 510) including the grouping of the views in each component window, and navigation links are specified between the component views for each component window (step 515). An interface view is specified for each component window (step 520). The interface view is the visual representation of the component. An embedded component can define more than one component window, where each component window has an associated interface view. An application or another component embeds one of the component windows by embedding the associated interface view.

The layout of the views in step 400 (FIG. 4) can be specified using pre-defined layout tools provided by an application development framework. Each pre-defined layout tool specifies a layout of view areas included in the pre-defined layout tool. Views can be grouped in step 400 by selecting a pre-defined layout tool and associating the views with specified view areas of the pre-defined layout tool. Examples of pre-defined layout tools include, a T-Layout tool (FIG. 6A), a T-Layout 90° tool (FIG. 6B), a T-Layout 180° tool (FIG. 6C), a T-Layout 270° tool (FIG. 6D), and a Grid-Layout tool (FIG. 6E).

Views can also be grouped in step 400 (FIG. 4) using view container UI elements. View container UI elements can contain other views. Views are grouped in a view container

UI element by specifying view areas within the view container and associating the views with the specified view areas.

FIG. 7 illustrates a visual interface of an example implementing a shopping application, in accordance with one aspect of the invention. The shopping application specifies a window 700 for the visual interface. The visual interface includes seven views, "Header" 730, "Welcome" 735, "Product Search" 740, "Product Selection" 745, "Shopping Basket" 750, "Order" 705, and "Order Confirmation" 710. The views "Order" 705 and "Order Confirmation" 710 are at the top level, i.e., they fill the complete viewing space in the window 700 when one of them is shown. "Order" 705 specifies three view areas 715, 720, and 725 and specifies a grouping of views embedded in each view area. "Product Selection" 745 and "Shopping Basket" 750 are interface views specified by an embedded component. The views 745 and 750 are used like any other views by the embedding shopping application, except for the fact that the view controllers for embedded component views are not accessible by the embedder. The black disks represent outbound plugs and the white disks represent inbound plugs for the views. Navigation links specify the transitions between the views. For example, the navigation link connecting "Shopping Basket" 750 to "Order Confirmation" 710 specifies an event to be triggered in order to display "Order Confirmation" 710.

The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The invention can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

Method steps of the invention can be performed by one or more programmable processors executing a computer program to perform functions of the invention by operating on input data and generating output. Method steps can also be performed by, and apparatus of the invention can be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in special purpose logic circuitry.

To provide for interaction with a user, the invention can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

The invention can be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an

implementation of the invention, or any combination of such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

The invention has been described in terms of particular embodiments. Other embodiments are within the scope of the following claims. For example, the steps of the invention can be performed in a different order and still achieve desirable results. What is claimed is: